Code Appendix A
Safe Path Generation and Robot Control GUI

Package(s): SafepathGUI

Description:

This code package is the main GUI for interfacing with the EyeRobot1. It retrieves data from the robot and allows the user to record the robot's position in either Cartesian coordinates or in terms of the actuator counts on each of the robot's 5 joints. It additionally will store all data points in a text file, which can be later imported into Matlab or other software for analysis. Finally, this program allows the user to command to the robot to move to a recorded safe path position.

NOTE: This program is still a work in progress. There are certain features which do NOT work as of May 18, 2011. These include the ability to move the robot via Joint Angles, and the ability to accurately move using Cartesian coordinates. The former can likely be resolved by more analysis of how the EyeRobot actuators work. The latter issue is due to the fact that the inverse kinematics of EyeRobot1 are not fully developed.

It is our recommendation that this only be used as a PROOF-OF-CONCEPT until it is further developed.

The GUI is developed using FLTK and can be edited using FLUID, a cross-platform interface editor.

Platforms Tested:

Ubuntu 10.10 (connected to EyeRobot via Ethernet)

Requirements:

The CISST library (at least version 2.4) with cisstStereoVision installed.
ZeroC ICE for communication to the EyeRobot

Installation:

This package can be compiled via CMake.

Screenshots

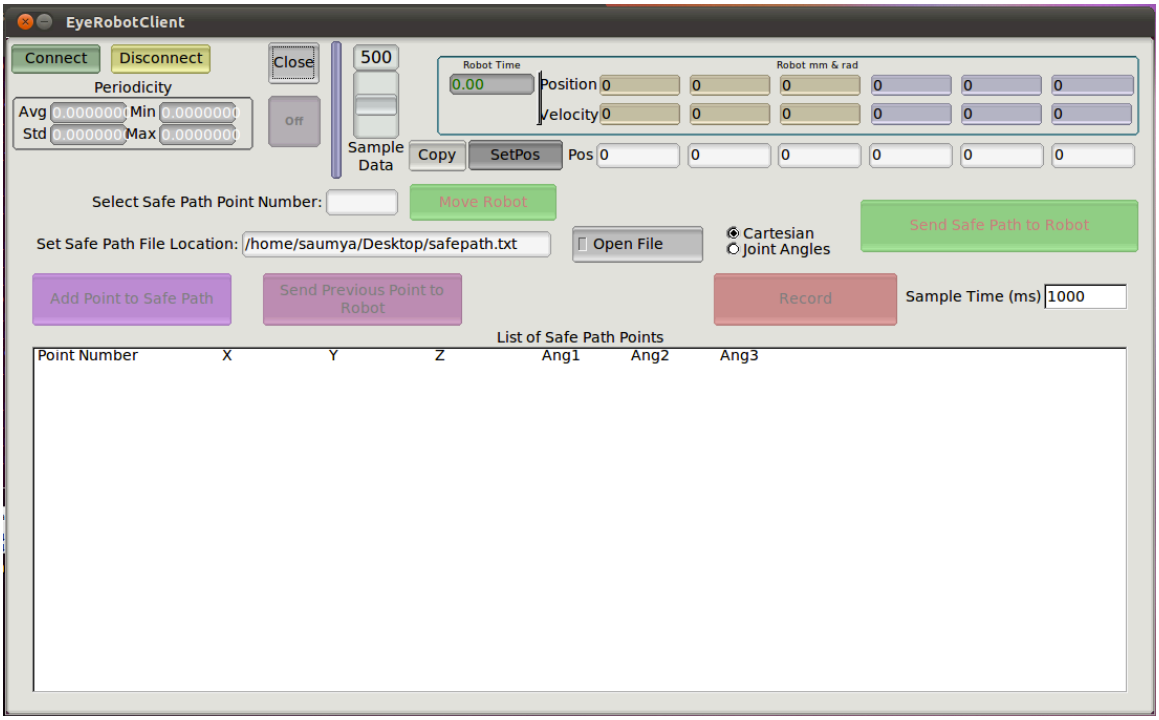

Figure 1—The GUI interface.

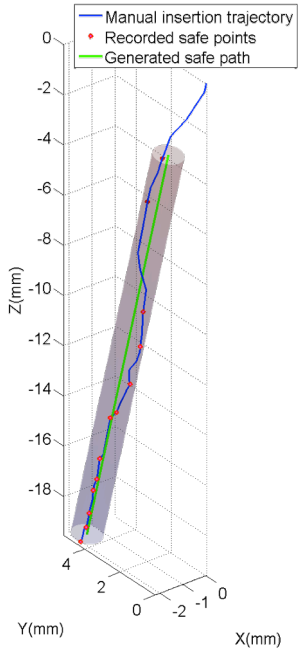

Figure 2 – Image (created in MATLAB) using safe point data recorded by our program. The cylinder represents a 1mm diameter virtual fixture that we would ideally keep the borescope in.

Future Work

As mentioned, this program has not fully been developed yet. The program works for manual safe path generation, in the sense that it can record the Robot's position in Cartesian coordinates and in Joint actuator counts. It can also record the position repeatedly at a set rate while an operator guides the borescope via the EyeRobot. Finally, the export to a text file works without issue. Thus, a manual recording and generation of safe paths is possible.

Future work includes automation of safe path generation. This requires incorporation with other components of our project: segmentation for detecting where the target window is; optical flow for depth perception; and 3D reconstruction to produce a mesh to allow for safe path and virtual fixture creation.

More immediately, work needs to be done to fix the ability to send safe path commands back to the robot. While we are able to send target Cartesian coordinates to EyeRobot1, the current inverse kinematics are not implemented properly. The robot will often move to a position a few millimeters away from the target, an error which is much too high for testing even in a cochlear phantom.

Using joint angles as a method of recording and measuring safe paths is a way to avoid inverse kinematics, since only forward kinematics are needed to get the real world position of the robot. However, sending these back to the robot and commanding it to move have not been implemented, and will require further analysis of the CISST Robot API.

The source code for this GUI and safe path generation are heavily documented with inline comments, so as to make it relatively easy for future developers to work on building this project further.

Source Code for the C++ files is attached. The code for the FLTK files is not included here, because it is long and does not add significant meaning to this report.